Networks problem sheet 1

Aleksander Kaminski

January 22, 2021

1 Introduction

(1.a): I've already got some programming experience in matlab which guided my decision to never use it again. I will choose to use python, as I don't know it yet.

(1.b): I primarily used website 'learnxiny' [1]to learn the syntax. This was enough for me as a reference to get started. Any other problems I had I consulted the python docs.[2]

(1.d): The format (gml) seems to simply be a text file, firstly defining the nodes and their attributes (label, id) and then a definition of the edges, with edge attributes. The power network didn't have any labels, so I had to manually tell networkx.read_gml to take the id as labels. I used the US Power grid dataset[1], and the Les Misérables[2] dataset. I shall call these power and lesmis, respectively.

(1.e): Firstly, the lesmis dataset, I generated a graph visualisation centered on the node with the highest degree, as shown in Fig. 1. As expected, this node represented the main character of the book. Fig. 2 shows that the second-highest degree nodes had significantly fewer connections than the highest, further showing that the highest-degree character appears with a much more varied audience.

For the power dataset, I firstly used python and networkx to generate a csv file with the degree distribution, then plotted it using R with ggplot2. I also used R to analyze the data. Firstly, I plotted the log-log graphs as shown in Fig. 4. This fit wasn't linear enough to deem correct, and so I moved on to a log- plot as in Fig. 5, which clearly showed a linear behaviour in the subset of degrees [2, 14]. From this, I extracted the exponential behaviour as $\exp(8.25 - 0.528 * degree)$. Fig. 6 shows that there are a large number of "leaf" nodes, namely, nodes which only connect to one other node.

[1]https://learnxinyminutes.com/docs/python/
[2]https://docs.python.org/3/c-api/set.html

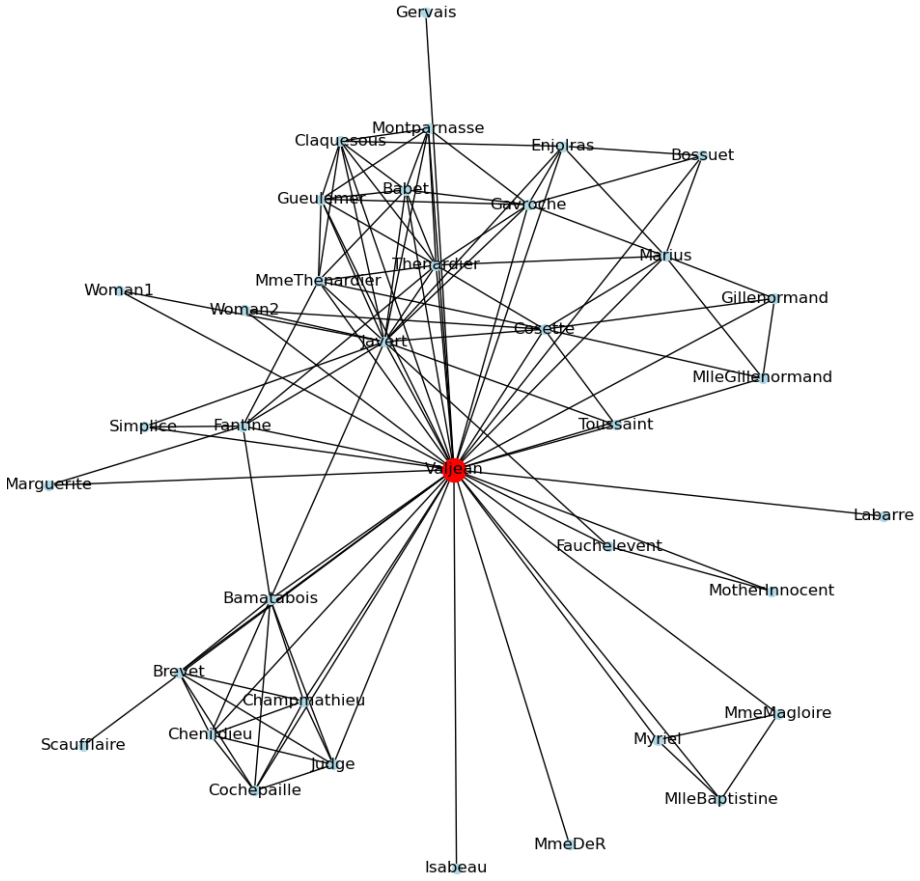

Figure 1: lesmis ego graph

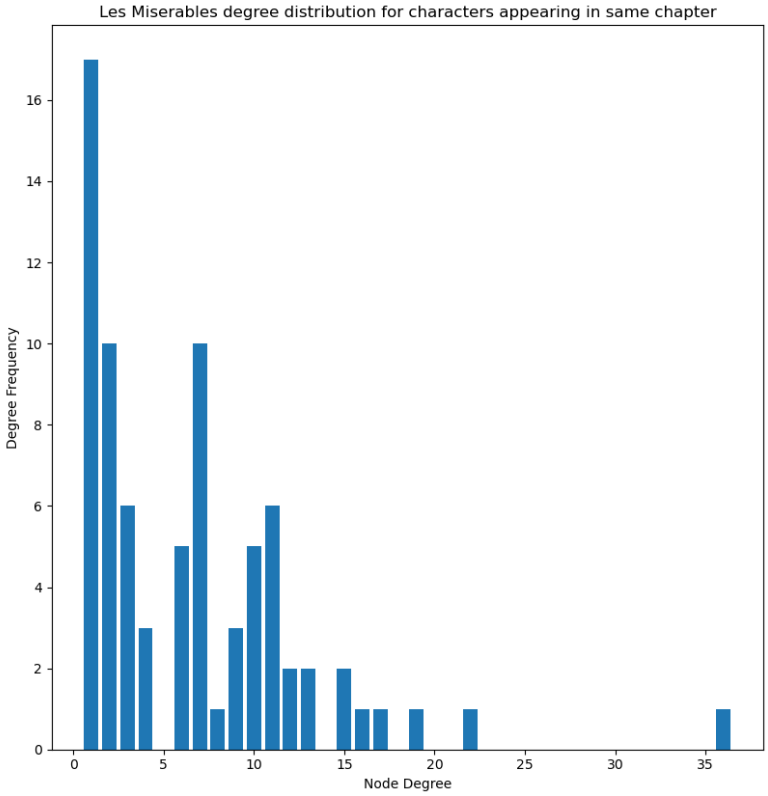

Figure 2: lesmis degree distribution

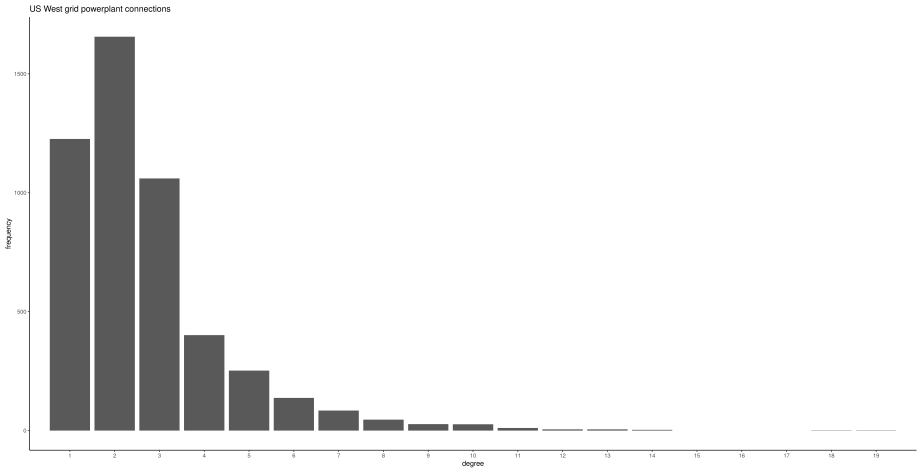

Figure 3: power degree distribution

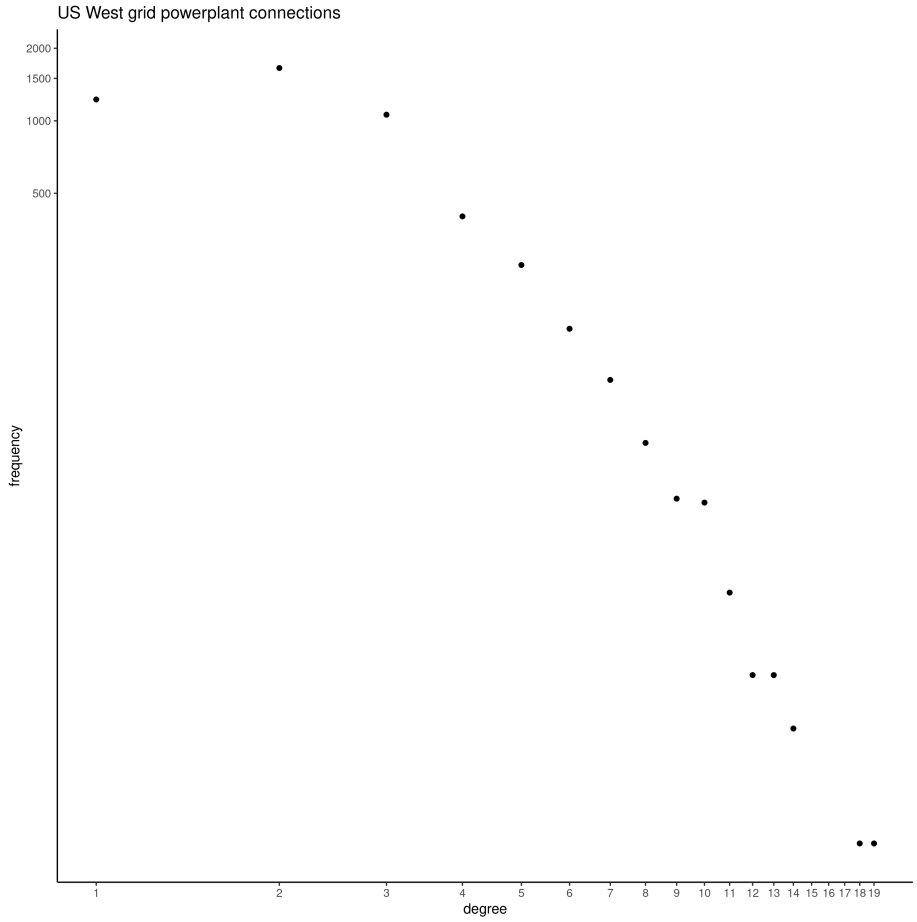

Figure 4: power degree distribution

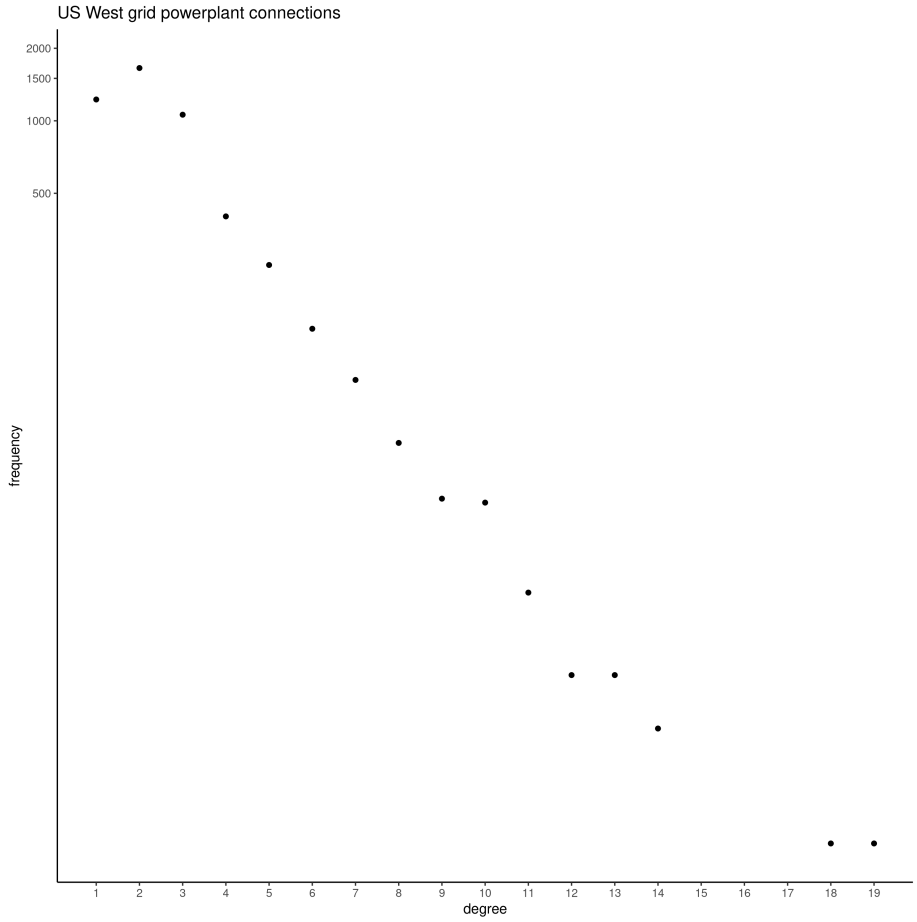

Figure 5: power degree distribution

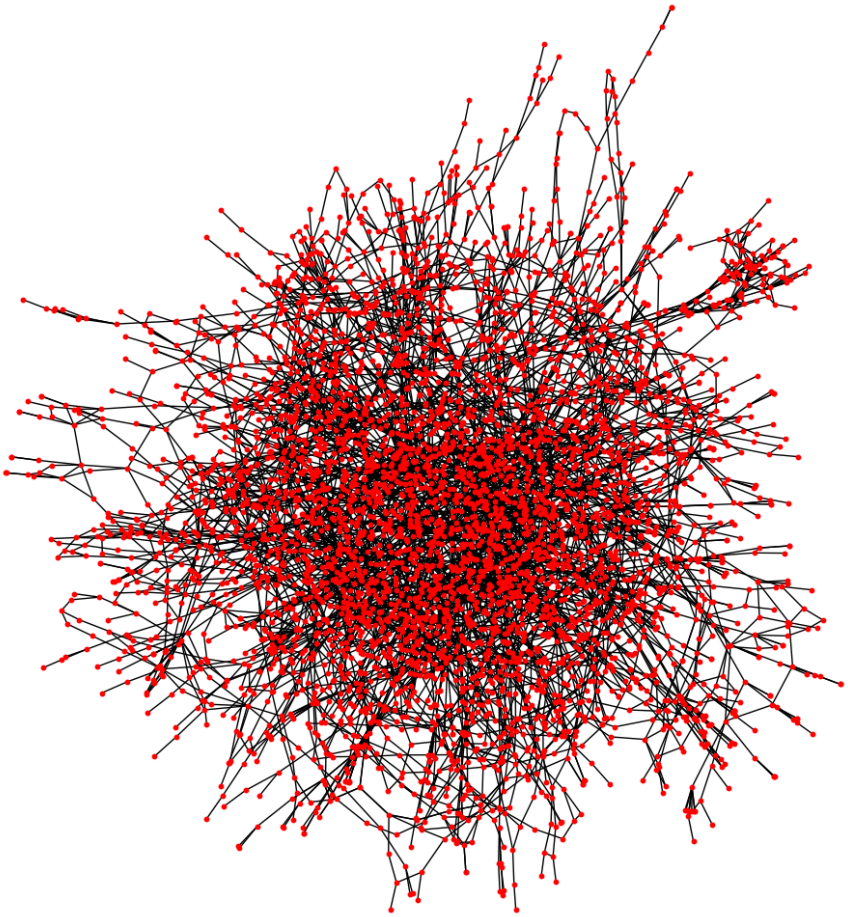

Figure 6: power network visualization

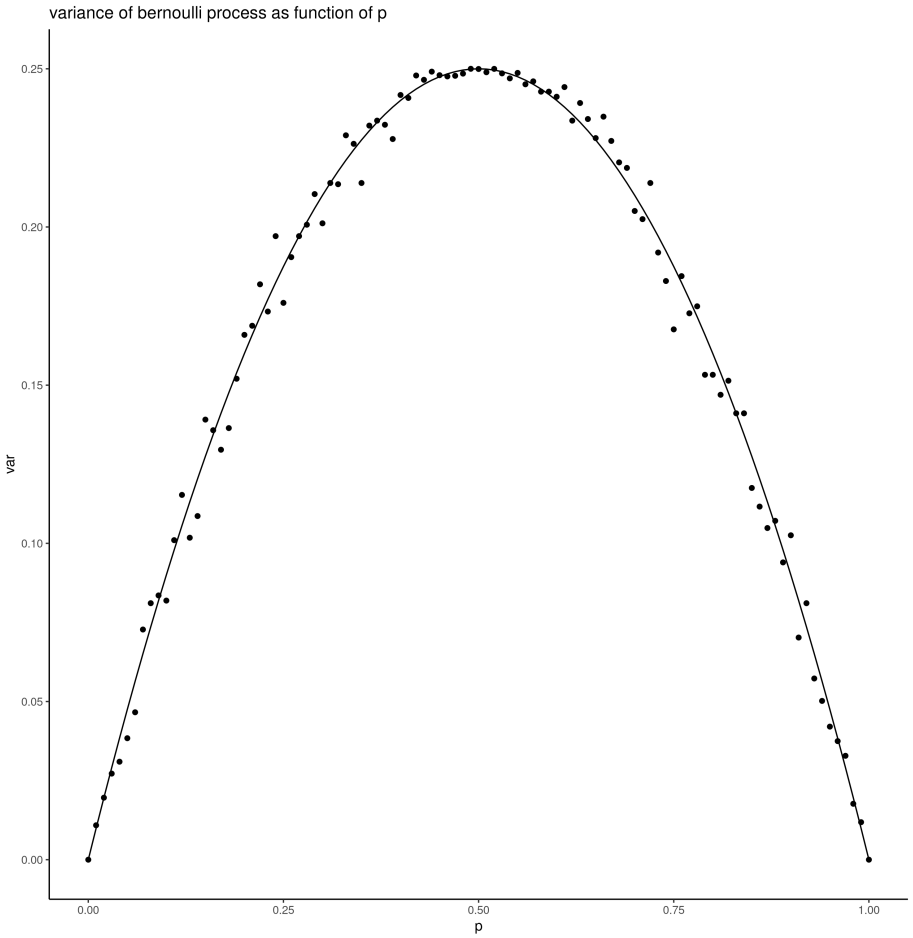

Figure 7: bernoulli process variance, with overlayed $p(1-p)$:

2 Mathematical Toolbox

(2. a) Fig. 7 shows that the computed variance matches the mathematics. The mean was computed to be p.

(2. b) I simulated 10000 random walkers and plotted the resulting position distribution for 5000, 10000 and 15000 samples. From theory (the notes), we know the data should follow the gaussian profile

$$p(x;t) = \frac{1}{\sqrt{2\pi Dt}} \exp \frac{-x^2}{4Dt}.$$

With D given by,

$$D = \frac{\langle x^2 \rangle}{2} = \frac{1}{2}.$$

With this, the data didn't fit, as the gaussian is not normalized properly. The correct normalisation should be, for continuous data $\frac{1}{\sqrt{4\pi Dt}}$. (Is this a

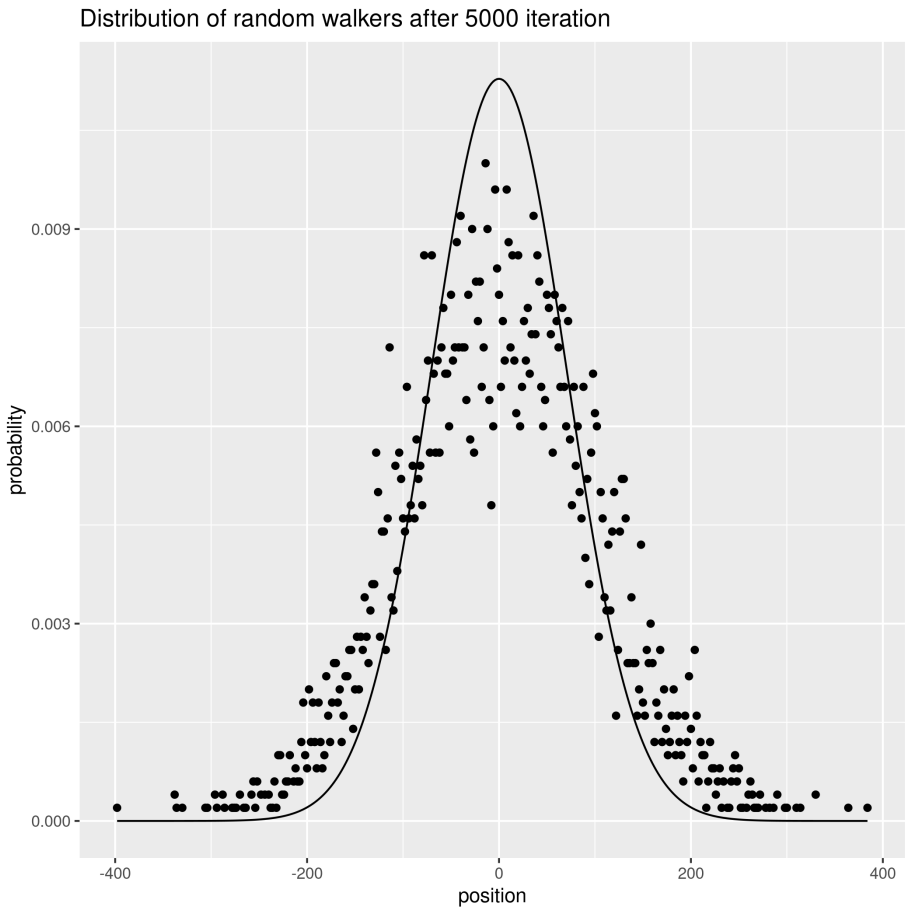

Figure 8: random walk after 5000 iterations

problem with the notes?) With this normalisation, our gaussian is at half of the amplitude of the data, and that's because at any given iteration, we can only access either the even-, or odd-numbered positions. Thus, we are overcompensating and need to multiply by two. Thus, for our data, the correct normalisation is

$$p(x;t) = \frac{1}{\sqrt{\pi Dt}} \exp \frac{-x^2}{4Dt}.$$

Figs. 8, 9, 10 show the data fitting to the modified gaussian correcly.

(2. c) Generating the matrices I found the distribution shown in Figs. 11, 12. The distribution looks like a semi-circle.

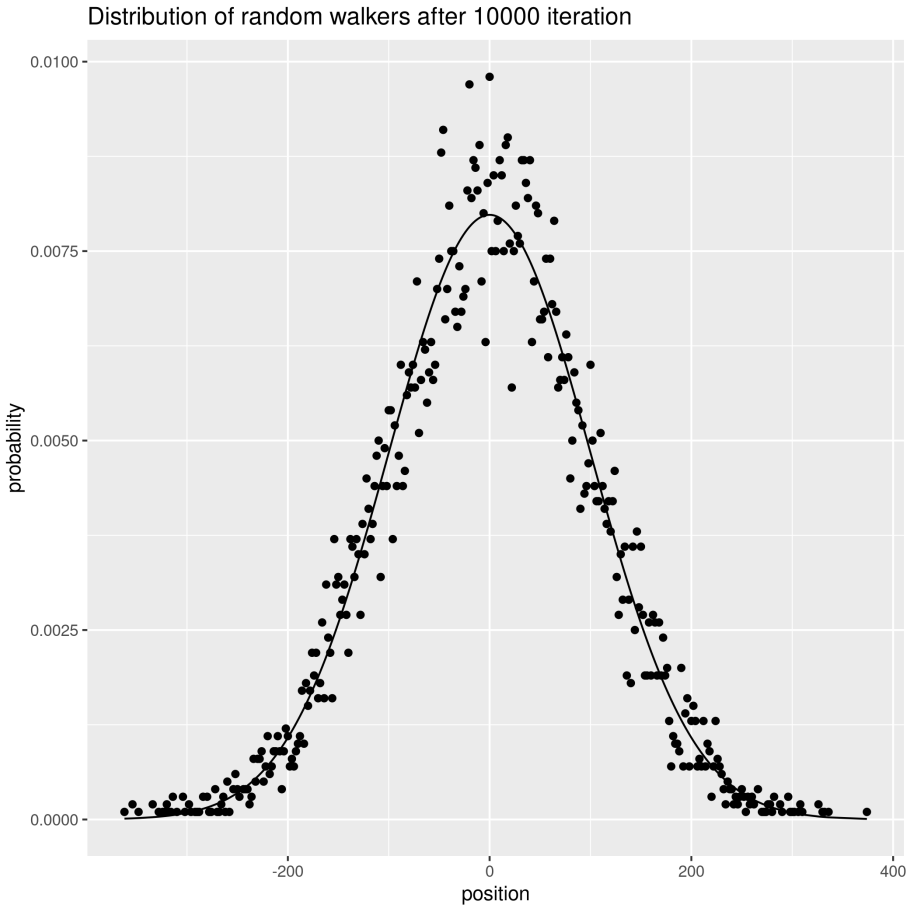

Figure 9: random walk after 10000 iterations

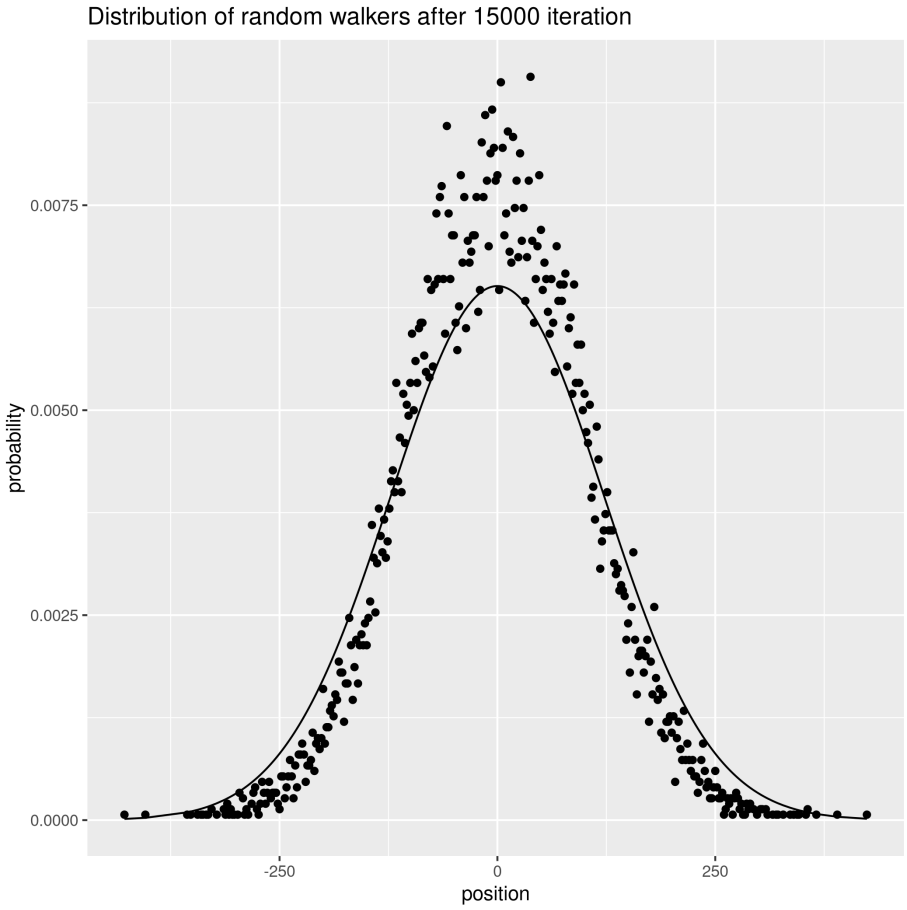

Figure 10: random walk after 15000 iterations

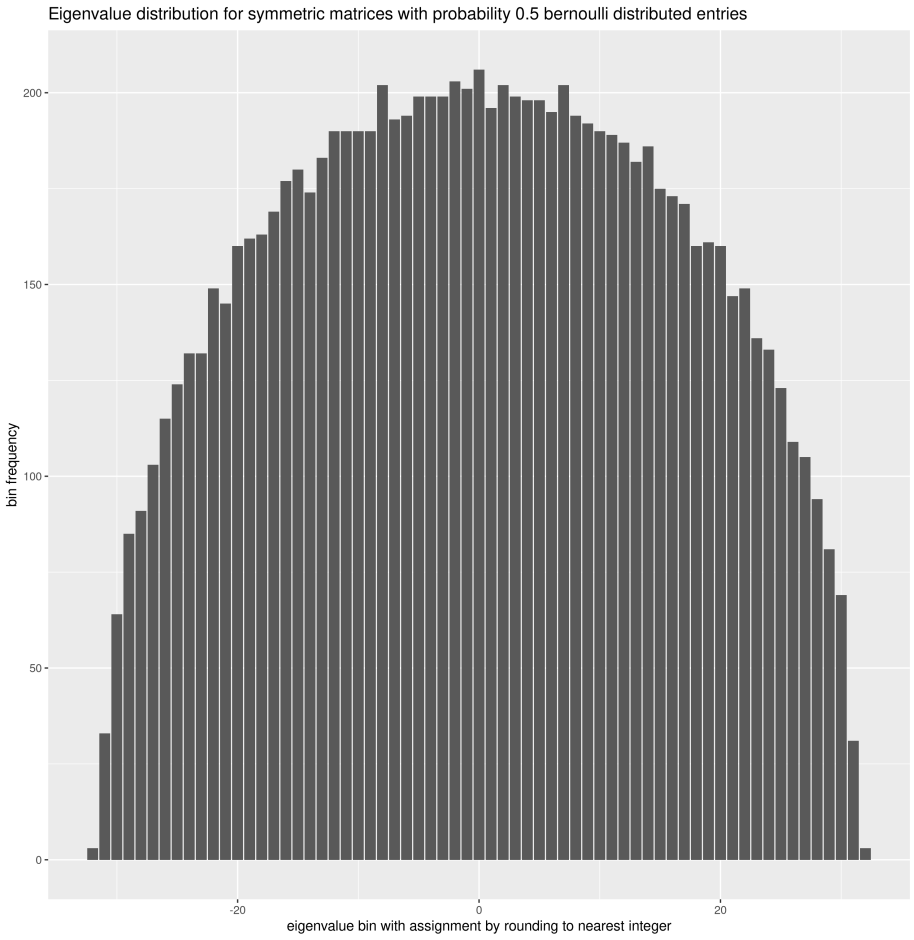

Figure 11: Eigenvalue distribution for bin < 100

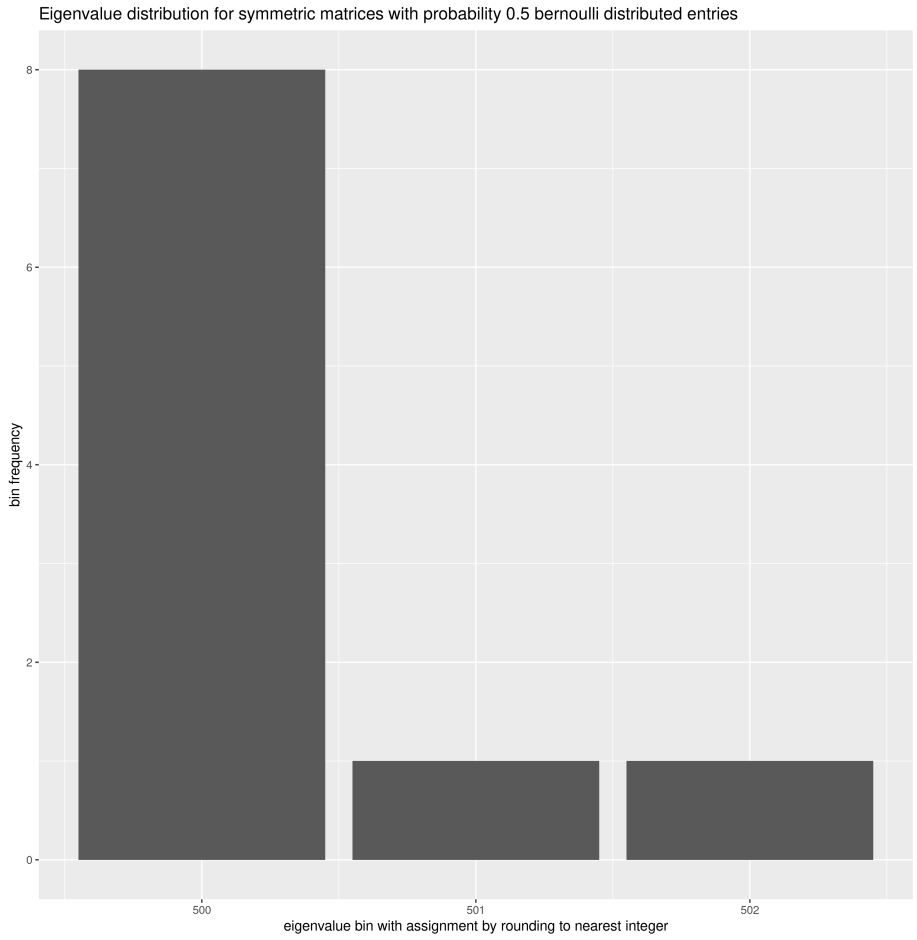

Figure 12: Eigenvalue distribution for bin > 100

3 Connectedness

(3. a) Not strongly connected implies that for some node i, there exists a node that cannot be reached by a walk. Thus, there will be a set of nodes with an asymptotic density of walkers equal to zero.
As an example, consider a strongly connected graph, g, with the addition of one directed node pointing into a node of g. Random walkers starting inside g will be contained there, and any walkers starting on the node outside g will soon end up inside. Thus, g will act as a sort of trap for walkers.

(3. b) The difference between directed and undirected graphs is in the symmetry of the adjacency matrix. We know from liner algebra that hermitian matrices (and thus, more weakly, symmetric matrices) have real eigenvalues. Thus, we would expect the eigenvalues of an undirected graph to generally be complex.

4 Clustering Coefficients

(4. a) I generated a network of size 10 given by the adjacency matrix

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

This network can be seen in Fig. 13, where the node labels are the local clustering coefficient. For this network, the global clustering coefficient is 0.473.

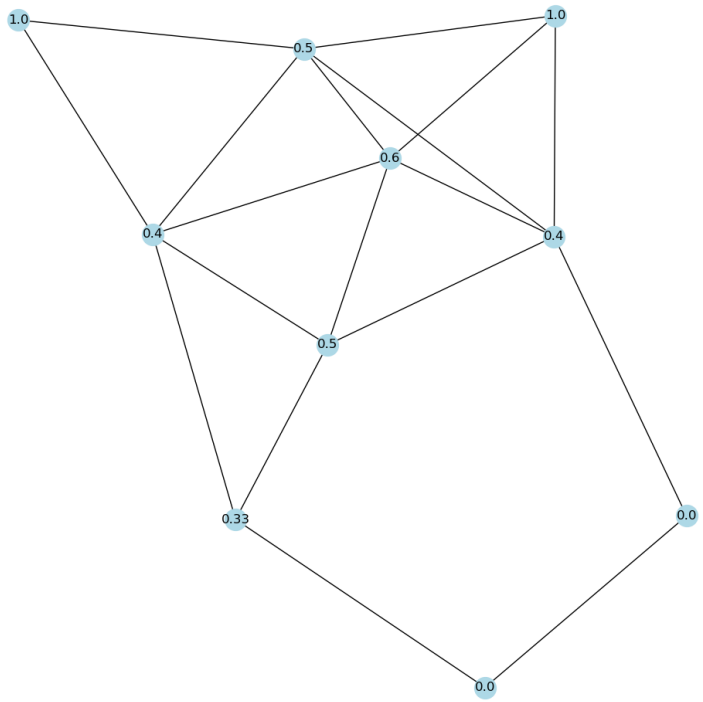

Figure 13: Random network with varying local clustering coefficients

5 Small World

(5. a) Firstly, for a network of size N, with connectivity, k, the maximum number of shortcuts is

$$S_{max} = \frac{(N-k)(N-k-1)}{2} - \frac{k(k+1)}{2}.$$

which can be seen by calculating the number of zeros in the adjacency matrix.
Then, for $N = 20$ and $k = 3$, I calculated diameters and clustering coefficient averaged over 10 runs. The data can be seen in Fig. 14. The diameter plateaus at integer values, which is to be expected. Naively, I would predict the length of each plateau to be roughly N, but this is not the case here for D=2.

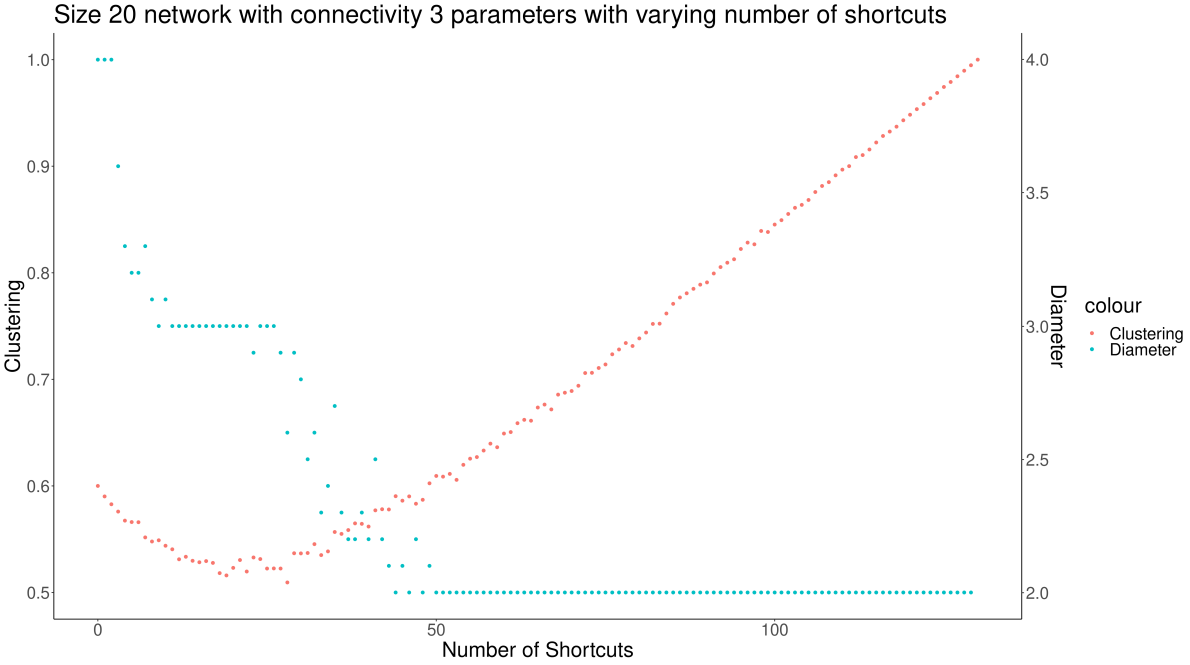

Figure 14: Clustering coefficient and diameter of a lattice-network with 20 nodes, and 6 neighbors

6 Centrality measures

(6. a) I generated undirected networks with $N = 20$ with varying densities, p, defined as the ratio of off-diagonal 1s to zeroes of the adjacency matrix. For each such network, I calculated the distributions of centrality measures:

(a) Degree centrality

(b) Betweenness Centrality

(c) Closeness Centrality

(d) Katz Centrality[3]

For each pair of centralities, I then calculated the Pearson correlation. The data is shown in Fig. 15. Correlations peak at $p = 0$ and $p = 1$, as here every node will have the same value of centrality. Only at the beginning, with very low density of connections, are the centralities correlated. All of the centralities are highly uncorrelated with Katz centrality, for most densities.

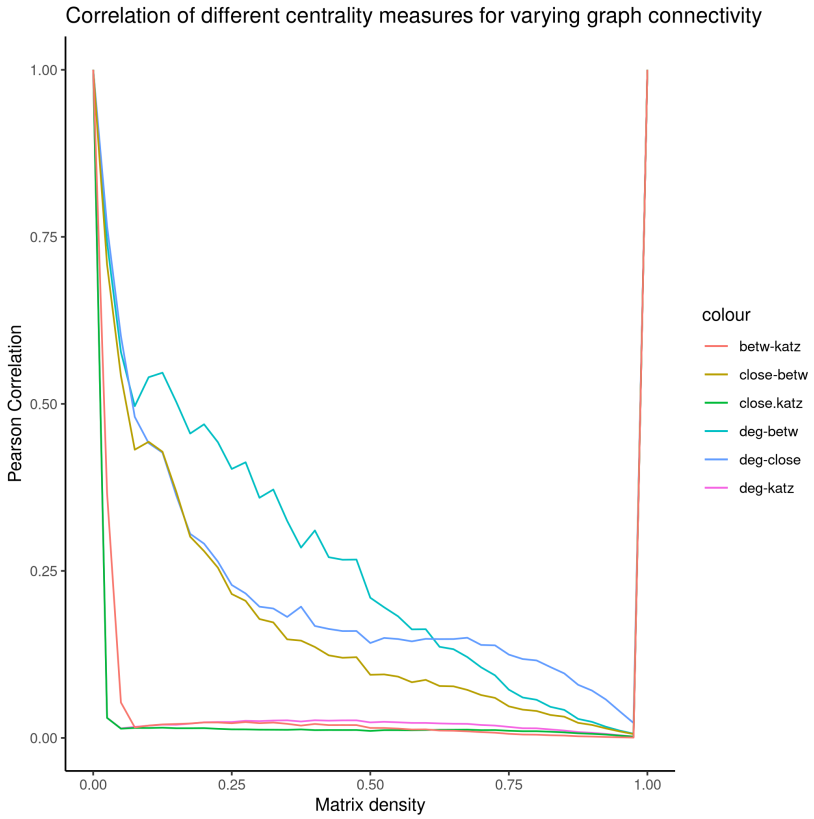

Figure 15: Centrality Correlations for $N = 20$ graphs

[3]I used an alpha parameter of 0.01 to assure convergence

(6. b) Assuming the same notation as the paper, with $Au = \lambda u$ and $\nu^T A = \nu^T \lambda$, and differences $A \mapsto A + \Delta A$, $\lambda \mapsto \lambda + \Delta\lambda$, $\nu^T \mapsto \nu^T + \Delta\nu^T$, $u \mapsto u + \Delta u$. Then, ignoring second order terms and assuming all differences are small,

$$\nu^T(A + \Delta A)(u + \Delta u) = \nu^T(\lambda + \Delta\lambda)(u + \Delta u).$$
$$\nu^T(Au + A\Delta u + \Delta A u) = \nu^T(\lambda u + \lambda\Delta u + \Delta\lambda u).$$
$$\nu^T\lambda\Delta u + \nu^T\Delta A u = \nu^T\lambda\Delta u + \nu^T\Delta\lambda u.$$
$$\Delta\lambda = \frac{\nu^T\Delta A u}{\nu^T u}.$$

For the removal of a single edge $i \to j$, we need only remove the corresponding entry in the matrix. So, $(\Delta A)_{ij} = -A_{lm}\delta_{il}\delta_{jm}$ and thus, we get that

$$\hat{I}_{ij} = -\frac{\Delta\lambda_{ij}}{\lambda} = \frac{A_{ij}\nu_i u_j}{\nu^T u}.$$

For the case of removal of a whole node i, we cannot assume that Δu is small, as the new u will have a 0 at i position. We can set $\Delta u = \delta u - u_i\hat{e}_i$, where δu_k are the small variations in the $k \neq i$ components. Removing a node removes the whole row and column from the matrix, and so the perturbation is given by $(\Delta A)_{kl} = -A_{kl}(\delta_{ki} + \delta_{li})$. Substituting Δu, the dynamical importance of a node is given by,

$$\hat{I}_i = -\frac{\Delta\lambda_i}{\lambda} = -\frac{\nu^T\Delta A u - u_i\nu^T\Delta A\hat{e}_i}{\nu^T u - \nu_i u_i}.$$

Now using the expression for ΔA we can see that $\nu^T A u = -2\lambda u_i\nu_i$ and $u_i\nu^T\Delta A\hat{e}_i = \lambda u_i\nu_i$. Also, with a high enough network size, the denominator can be approximated as $\nu^T u - \nu_i u_i \approx \nu^T u$. Thus, we end up with

$$\hat{I}_i = \frac{\nu_i u_i}{\nu^T u}.$$

References

[1] H. Lietz, *Watts, Duncan J./Strogatz, Steven H. (1998). Collective Dynamics of » Small- World « Networks. Nature 393, S. 440 – 442.*, pp. 551–553. 01 2019.

[2] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," 2006.